Chapter 2

State Space Models – Basic Concepts

Related reading in Bay:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Subsection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Models of Linear Systems)</td>
<td>1.1</td>
<td>1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.5</td>
</tr>
<tr>
<td>1.2</td>
<td>1.2.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.2.2</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this Chapter we provide some fundamental results on state space systems which will be useful later on: canonical forms and changes of state coordinates. Also, we describe relations between transfer matrices and state space models and how to form interconnections of state space models. Finally we end up with a method to compute a linear approximation of a nonlinear system.

We recall the LTI state space model (1.4):

\[
\begin{align*}
\dot{x} &= Ax + Bu \\
y &= Cx + Du
\end{align*}
\]

(2.1a) (2.1b)

where \(u \in \mathbb{R}^m\), \(y \in \mathbb{R}^p\), \(x \in \mathbb{R}^n\).

### 2.1 State space to transfer function (ss2tf)

For a LTI state space system (2.1) we can take the Laplace transform of both sides of the state space equations (2.1) to obtain

\[
\begin{align*}
\mathcal{L}\{\dot{x}\} &= s\mathcal{L}\{x\} - x(0^-) = \mathcal{L}\{Ax\} + \mathcal{L}\{Bu\} = A\mathcal{L}\{x\} + B\mathcal{L}\{u\} \\
\mathcal{L}\{y\} &= C\mathcal{L}\{x\} + D\mathcal{L}\{u\}
\end{align*}
\]

(2.2a) (2.2b)
where

\[ L\{x\} = \begin{bmatrix} L\{x_1\} \\ \vdots \\ L\{x_n\} \end{bmatrix} \]

Also in (2.2) we have used the linearity property of the Laplace transform, i.e., \( L\{Ax\} = AL\{x\}, L\{Bu\} = BL\{u\}, L\{Cx\} = CL\{x\}, L\{Du\} = DL\{u\} \).

We denote \( L\{x\} = X(s), L\{u\} = U(s), L\{y\} = Y(s) \) as the Laplace transform of vectors \( x, u, \) and \( y \) respectively, and assuming \( x(0^-) = 0 \), as is customary for transfer function models, we have from (2.2)

\[
\begin{align*}
(sI - A)X(s) &= BU(s) \quad (2.3a) \\
Y(s) &= CX(s) + DU(s) \quad (2.3b)
\end{align*}
\]

Solving for \( X(s) \) in (2.3a) gives \( X(s) = (sI - A)^{-1}BU(s) \) which can be substituted into (2.3b) to obtain

\[
Y(s) = (C(sI - A)^{-1}B + D)U(s) = G(s)U(s) \quad (2.4)
\]

where \( G(s) = (C(sI - A)^{-1}B + D) \) is the \( p \times m \) transfer matrix from \( u \) to \( y \). We remark that if we expand \( B = [B_1, \ldots, B_m], B_i \in \mathbb{R}^{n \times 1} \) into its columns and \( C^T = [C_1^T, \ldots, C_p^T], C_i \in \mathbb{R}^{1 \times n} \) into its rows we have

\[
G(s) = \begin{bmatrix} C_1 \\ \vdots \\ C_p \end{bmatrix} (sI - A)^{-1} \begin{bmatrix} B_1 & \ldots & B_m \end{bmatrix} + D
\]

Hence, the \((i, j)\)th entry of \( G(s) \) is given by

\[
G_{ij}(s) = C_i(sI - A)^{-1}B_j + D_{ij}
\]

where \( D_{ij} \) is the \((i, j)\)th entry of \( D \).

We recall Cramer’s rule for matrix inverse [Bay99, pg. 518]:

\[
A^{-1} = \frac{\text{adj}(A)}{\text{det}(A)}
\]

where \( \text{adj}(A) \) is adjoint of \( A \) and \( \text{det}(A) \) is the determinant of \( A \). The \((i, j)\)th element of the adjoint matrix of \( A \) is the cofactor of \((j, i)\)th element of \( A \). The cofactor, or “signed minor”, is the \((-1)^{i+j}\) times the determinant of the submatrix obtained by deleting the row and column containing the \((i, j)\)th element of \( A \). For example for \( n = 2 \) we have

\[
A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \text{adj}(A) = \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}, \quad \text{det}(A) = a_{11}a_{22} - a_{21}a_{12}
\]
For \( n = 3 \) we have

\[
A = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
\end{bmatrix}, \quad \text{adj}(A) = \begin{bmatrix}
  a_{22} & a_{23} \\
  a_{32} & a_{33} \\
  a_{31} & a_{32}
\end{bmatrix} - \begin{bmatrix}
  a_{12} & a_{13} \\
  a_{32} & a_{33} \\
  a_{31} & a_{32}
\end{bmatrix} - \begin{bmatrix}
  a_{12} & a_{13} \\
  a_{23} & a_{33} \\
  a_{21} & a_{23}
\end{bmatrix}
\]

\[
det(A) = a_{11} \det \begin{bmatrix}
  a_{22} & a_{23} \\
  a_{32} & a_{33}
\end{bmatrix} - a_{12} \det \begin{bmatrix}
  a_{21} & a_{23} \\
  a_{31} & a_{33}
\end{bmatrix} + a_{13} \det \begin{bmatrix}
  a_{21} & a_{22} \\
  a_{31} & a_{32}
\end{bmatrix}
\]

(2.5)

where we have used \(|A|\) to denote \(\text{det}(A)\). Cramer’s rule is not an efficient method for computing a matrix inverse and should only be used for theoretical arguments.

From Cramer’s rule we can make the following observation about

\[
C(sI - A)^{-1}B = \frac{1}{\det(sI - A)} \text{adj}(sI - A)B
\]

We observe that since

\[
sI - A = \begin{bmatrix}
  s - a_{11} & -a_{12} & \cdots & -a_{1n} \\
  -a_{21} & s - a_{22} & \cdots & -a_{2n} \\
  \vdots & \vdots & \vdots & \vdots \\
  -a_{n1} & -a_{n2} & \cdots & s - a_{nn}
\end{bmatrix}
\]

so \(\det(sI - A)\), which is called the characteristic polynomial of \(A\), has degree\(^1\) \(n\) since its expansion has only one term which is the product of the diagonal elements of \(sI - A\):

\[
(s - a_{11})(s - a_{22}) \cdots (s - a_{nn})
\]

A similar argument shows that matrix \(\text{adj}(sI - A)\), whose entries are obtained by deleting a row and column from \(sI - A\) and computing a determinant, has entries which are polynomials of degree at most \(n - 1\). Hence, \(\frac{1}{\det(sI - A)} \text{adj}(sI - A)B\) is a strictly proper transfer matrix, i.e., a transfer matrix whose transfer function entries have denominator polynomials of degree strictly greater than their numerator degree. Therefore, when \(D = 0\), then \(G(s)\) is strictly proper. For state space systems with nonzero \(D\) matrix, the transfer matrix is proper but not strictly proper, i.e., there exists at least one entry of \(G(s)\) whose numerator degree equals its denominator degree. Note that a system with a state space model can not have improper entries in its transfer matrix.

We say \(\lambda\) is pole of a transfer matrix if it is a pole of any of its transfer function entries. Clearly, the poles of a transfer matrix are the roots of the characteristic equation of \(A\):

\(^1\)The degree (or sometimes “order”) of a univariate polynomial \(p(\lambda)\) of a scalar variable \(\lambda\) is given by its highest power. For example,

\[
p(\lambda) = a_0 + a_1\lambda + \cdots + a_{r-1}\lambda^{r-1} + \lambda^r.
\]

has a degree of \(r\). We denote this \(\deg(p) = r\).
\[
\det(sI - A) = 0 \text{ which are the eigenvalues of } A. \text{ We remark that an eigenvalue of } A \text{ may not be a pole of a transfer matrix if pole-zero cancelations occur. Chapter 3 covers the necessary background on linear algebra including eigenvalues.}
\]

From classical control we recall that for SISO systems \( z_i \) is a zero if \( G(z_i) = 0 \). For the MIMO case this notion is generalized into the so-called transmission zero \( z \) which satisfies

\[
G(z)w = 0
\]

for some \( w \neq 0 \). For \( p = m \) or square systems this condition can be expressed more conveniently as

\[
\det(\begin{bmatrix} -zI + A & B \\ C & D \end{bmatrix}) = 0. \tag{2.6}
\]

There are two ways to perform \texttt{ss2tf} in MATLAB:

- Using \([\text{num,den}]=\texttt{ss2tf} \ (A,B,C,D,iu)\) which computes the transfer matrix from input \( iu \) to the output. \texttt{num} has \( p \) rows for a \( p \)-output system. The \( i \)th row in \texttt{num} corresponds to the numerator polynomial for the \( i \)th component of the output. This requires numerical \( A, B, C, D \).

- Using the Symbolic Math Toolbox. This method does not required numerical \( A, B, C, D \) and can work with general MIMO systems. Hence, it is more general than \texttt{ss2tf}.

We remark that for a given state space model (2.1) we always obtain a corresponding unique transfer matrix model. We will see shortly that going from transfer matrix to state space does not share this uniqueness.

**Example 6** From the armature controlled DC motor in Example 5 when \( L_a = 0 \) we have

\[
\begin{align*}
\dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & -K_2 \end{bmatrix} x + \begin{bmatrix} 0 \\ K_1 \end{bmatrix} u \\
y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x
\end{align*} \tag{2.7}
\]

We have

\[
sI - A = \begin{bmatrix} s & -1 \\ 0 & s + K_2 \end{bmatrix} \tag{2.8}
\]

Taking the inverse we obtain

\[
(sI - A)^{-1} = \frac{1}{s(s + K_2)} \begin{bmatrix} s + K_2 & 1 \\ 0 & s \end{bmatrix} \tag{2.9}
\]

Hence we have the strictly proper transfer function model

\[
\frac{Y(s)}{U(s)} = G(s) = C(sI - A)^{-1}B = \frac{K_1}{s(s + K_2)} \tag{2.10}
\]
We remark that the eigenvalues of $A$ are 0 and $-K_2$ which are the poles of the transfer function. From (2.6)

$$\det\left(\begin{bmatrix} -zI + A & B \\ C & D \end{bmatrix}\right) = K_1 > 0$$

the system has no zeros.

**Example 7** We consider the mass spring system in Example 3. Although we could use (2.4) to compute the transfer matrix, this would involve a $4 \times 4$ matrix inverse which is unpleasant to perform by hand. Instead we take Laplace transforms of (1.5) and solve 2 linear equations for $Y(s)$ in terms of $U(s)$. This approach only requires a $2 \times 2$ matrix inverse.

From (1.5) we have

$$\begin{bmatrix} s^2m_1 + k_1 + k_2 & -k_2 \\ -k_2 & s^2m_2 + k_1 + k_2 \end{bmatrix} \begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}$$

Solving for $Y(s) = (Y_1(s), Y_2(s))^T$ gives

$$Y(s) = \frac{1}{(s^2m_1 + k_1 + k_2)(s^2m_2 + k_1 + k_2) - k_2^2} \begin{bmatrix} s^2m_2 + k_1 + k_2 & k_2 \\ k_2 & s^2m_1 + k_1 + k_2 \end{bmatrix} U(s)$$

Using (2.6) we find the system has no transmission zeros as

$$\det\left(\begin{bmatrix} -zI + A & B \\ C & D \end{bmatrix}\right) = \frac{1}{m_1m_2} > 0$$

The MATLAB code for Examples 6 and 7 is

```matlab
% Example of ss2tf using symbolic math toolbox and SS2TF
% Armature-voltage controlled DC motor with La=0

syms K1 K2 s
A=[0 1;0 -K2]; B=[0;K1]; C=[1 0]; D=0;

disp('Transfer function for Ex #1 (Symbolic Math Computation)')
G=simple(C*inv(s*eye(size(A))-A)*B+D); pretty(G)

disp('Transmission zero computation')
det([-z*eye(size(A))+A B;C D])

disp('Transmission zero computation')
det([-z*eye(size(A))+A B;C D])

% To use SS2TF we need to assign numerical values to K1 and K2. SS2TF does
% not accept symbolic A,B,C,D

A_v=subs(A,K2,40); B_v=subs(B,K1,74);
[num,den]=ss2tf(A_v,B_v,C,D,1);
```
disp('Transfer function for Ex #1 using SS2TF (K1= 74 rad/(Vs^-2), K2=40 (1/s))')
tf(num,den)

% Example of ss2tf using symbolic math toolbox
% 2-DOF mechanical system

syms k1 k2 m1 m2
A=[0 1 0 0;-(k1+k2)/m1 0 k2/m1 0;0 0 0 1;k2/m2 0 -(k1+k2)/m2 0];
B=[0 0;1/m1 0;0 0;0 1/m2];
C=[1 0 0 0;0 0 1 0];
D=zeros(2);

disp('Transfer matrix for mass-spring system (Symbolic Math Computation)')
pretty(simple(C*inv(s*eye(size(A))-A)*B+D))

disp('Transmission zero computation')
det([-z*eye(size(A))+A B;C D])

2.2 Transfer function to state space (tf2ss)

For a given transfer matrix $G(s)$, if a LTI state space form (2.1) can be found (i.e., some $A, B, C, D$ matrices) so that $G(s) = C(sI - A)^{-1}B + D$, we say this state space model is a realization of $G(s)$. There are an infinite number of realizations for a given transfer matrix and these realization do not even have to have the same state dimension. Here we consider three particularly useful state space realizations for SISO LTI systems of the form (2.1) called the Controllable Canonical Form (CCF), Observable Canonical Form (OCF), and Jordan Canonical Form (JCF). The CCF and OCF will be especially useful when discussing state feedback and state observers later. A realization is said to be minimal if the dimension of the state variable is the smallest possible. It turns out that a realization is minimal iff its dimension is equal to the degree of the denominator polynomial of the system’s transfer function. This implies that transfer functions where a pole zero cancelation occurs do not have minimal realizations.

The Controllable Canonical Form (CCF)

The development starts with the simpler SISO system with no zeros

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0} \quad (2.11)$$

which can be written in time domain as

$$\frac{d^n y}{dt^n} = -\alpha_{n-1} \frac{d^{n-1} y}{dt^{n-1}} - \cdots - \alpha_1 \frac{dy}{dt} - \alpha_0 y + u \quad (2.12)$$
The simulation diagram of this system is given in Figure 2.1. We define the state variables as the output of the integrators:

\[
\begin{align*}
x_1 &= y \\
x_2 &= \dot{y} \\
\vdots \\
x_n &= y^{(n-1)}
\end{align*}
\]

When states are defined as successive time derivatives of a single system variable, the output \(y\) in this case, they are sometimes referred to as *phase variables*. In general, as we see below, states cannot be defined in this simple manner (see PS1 Q1). Time differentiating \(x_k\) defined in (2.13) and using (2.12) we obtain

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
\vdots \\
\dot{x}_{n-1} &= x_n \\
\dot{x}_n &= -\alpha_0 x_1 - \alpha_1 x_2 - \cdots - \alpha_{n-1} x_n + u \\
y &= x_1
\end{align*}
\]

or in matrix form (2.1):

\[
\begin{align*}
\dot{x} &= 
\begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-\alpha_0 & -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{n-1}
\end{bmatrix} x + 
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix} u
\end{align*}
\]

\[
y = 
\begin{bmatrix}
1 & 0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix} A \end{bmatrix} x
\]

Now we can consider the general proper SISO system

\[
G(s) = \frac{Y(s)}{U(s)} = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \cdots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0}
\]

we split the transfer function into two factors

\[
G(s) = \frac{Y(s)}{U(s)} = \frac{Y(s)}{Z(s)} \frac{Z(s)}{U(s)}
\]

where

\[
\frac{Z(s)}{U(s)} = \frac{1}{s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0}, \quad \frac{Y(s)}{Z(s)} = \frac{\beta_n s^n + \beta_{n-1} s^{n-1} + \cdots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0}
\]
We have already computed a state space realization for \(Z(s)\) in (2.14) and since

\[
y = \beta_n \frac{d^n z}{dt^n} + \beta_{n-1} \frac{d^{n-1} z}{dt^{n-1}} + \cdots + \beta_1 \frac{dz}{dt} + \beta_0 z
\]

we obtain Figure 2.2. Performing some block diagram manipulation which cancel the integrators and differentiators we obtain the simulation diagram in Figure 2.3. From Figure 2.3 the output is given by

\[
y = [\beta_0 - \alpha_0 \beta_n \beta_1 - \alpha_1 \beta_n \cdots \beta_{n-1} - \alpha_{n-1} \beta_n] x + \beta_n u
\]

Hence a state space realization of (2.15) is

\[
\dot{x} = 
\begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-\alpha_0 & -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{n-1}
\end{bmatrix}
\begin{bmatrix} x \end{bmatrix} + 
\begin{bmatrix} 0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}
\begin{bmatrix} u \end{bmatrix}
\]

\[
y = \left[\beta_0 - \alpha_0 \beta_n \beta_1 - \alpha_1 \beta_n \cdots \beta_{n-1} - \alpha_{n-1} \beta_n\right] x + \beta_n u
\]

Realization (2.16) is called the Controllable Canonical Form (CCF).

We remark that if we had defined \(\bar{x}_k = y^{(n-k)}, 1 \leq k \leq n\) which corresponds to taking the integrator outputs as states, but defines \(x_1, x_2, \ldots\) from left to right in Figure 2.3. This
2.2. TRANSFER FUNCTION TO STATE SPACE (TF2SS)

Definition of state yields the realization

\[
\begin{align*}
\dot{x} &= \begin{bmatrix}
-\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_{2} & -\alpha_{1} & -\alpha_{0} \\
1 & 0 & \cdots & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix} x_{1} \\ \vdots \\ x_{n-1} \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix} u \\
y &= \begin{bmatrix} \beta_{n-1} - \alpha_{n-1} \beta_{n} & \cdots & \beta_{1} - \alpha_{1} \beta_{n} & \beta_{0} - \alpha_{0} \beta_{n} \end{bmatrix} \begin{bmatrix} x_{1} \\ \vdots \\ x_{n-1} \end{bmatrix} + \beta_{n} u
\end{align*}
\]

Figure 2.2: Block diagram of (2.15). Summing junctions which are not labelled are positive.

Although the MIMO case is more involved, generating a SIMO CCF is a straightforward generalization of the SISO case. For example if we have a two output transfer matrix

\[
G(s) = \frac{1}{s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_{1}s + \alpha_0} \begin{bmatrix} \beta_{1n}s^n + \beta_{1n-1}s^{n-1} + \cdots + \beta_{11}s + \beta_{10} \\
\beta_{2n}s^n + \beta_{2n-1}s^{n-1} + \cdots + \beta_{21}s + \beta_{20} \end{bmatrix}
\]

we get

\[
\begin{align*}
\dot{x} &= \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-\alpha_{0} & -\alpha_{1} & -\alpha_{2} & \cdots & -\alpha_{n-1} \end{bmatrix} x + \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} u \\
y &= \begin{bmatrix} \beta_{10} - \alpha_0 \beta_{1n} & \beta_{11} - \alpha_1 \beta_{1n} & \cdots & \beta_{1n-1} - \alpha_{n-1} \beta_{1n} \\
\beta_{20} - \alpha_0 \beta_{2n} & \beta_{21} - \alpha_1 \beta_{2n} & \cdots & \beta_{2n-1} - \alpha_{n-1} \beta_{2n} \end{bmatrix} x + \begin{bmatrix} 0 \\ \vdots \end{bmatrix} u
\end{align*}
\]

This idea easily generalizes to the \( p > 2 \) output case.
The Observable Canonical Form (OCF)

Consider again a SISO system with transfer function (2.15). For this system we have

\[(s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0)Y(s) = (\beta_n s^n + \beta_{n-1} s^{n-1} + \cdots + \beta_1 s + \beta_0)U(s)\]

or

\[s^n(Y(s) - \beta_n U(s)) + s^{n-1}(\alpha_{n-1} Y(s) - \beta_{n-1} U(s)) + \cdots + (\alpha_0 Y(s) - \beta_0 U(s)) = 0\]

Dividing by \(s^n\) and solving for \(Y(s)\) gives

\[Y(s) = \beta_n U(s) + \frac{1}{s} (\beta_{n-1} U(s) - \alpha_{n-1} Y(s)) + \cdots + \frac{1}{s^n} (\beta_0 U(s) - \alpha_0 Y(s))\]

which has the simulation diagram shown in Figure 2.4. Defining states as the outputs of the integrators as in Figure 2.4 gives the state space equations

\[
\begin{align*}
\dot{x}_1 &= x_2 + \beta_{n-1} u - \alpha_{n-1} y = x_2 - \alpha_{n-1} x_1 + (\beta_{n-1} - \alpha_{n-1} \beta_n) u \\
\dot{x}_2 &= x_3 + \beta_{n-2} u - \alpha_{n-2} y = x_3 - \alpha_{n-2} x_1 + (\beta_{n-2} - \alpha_{n-2} \beta_n) u \\
&\vdots \\
\dot{x}_{n-1} &= x_n - \alpha_1 x_1 + (\beta_1 - \alpha_1 \beta_n) u \\
\dot{x}_n &= -\alpha_0 x_1 + (\beta_0 - \alpha_0 \beta_n) u \\
y &= x_1 + \beta_n u
\end{align*}
\]
2.2. TRANSFER FUNCTION TO STATE SPACE (TF2SS)

In matrix form we have

\[
\dot{x} = \begin{bmatrix}
-\alpha_{n-1} & 1 & 0 & \cdots & 0 \\
-\alpha_{n-2} & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\alpha_1 & 0 & 0 & \cdots & 1 \\
-\alpha_0 & 0 & 0 & \cdots & 0
\end{bmatrix} x + \begin{bmatrix}
\beta_{n-1} - \alpha_{n-1}\beta_n \\
\beta_{n-2} - \alpha_{n-2}\beta_n \\
\vdots \\
\beta_1 - \alpha_1\beta_n \\
\beta_0 - \alpha_0\beta_n
\end{bmatrix} u
\]

\[
y = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0
\end{bmatrix} x + \begin{bmatrix}
\beta_n
\end{bmatrix} u
\]

(2.18)

Realization (2.18) is called the Observable Canonical Form (OCF).

We remark that if we had defined \( \bar{x}_k = y^{(n-k)}, 1 \leq k \leq n \) which corresponds to taking the integrator outputs as states, but defines \( x_1, x_2, \ldots \) from left to right in Figure 2.4. This definition of state yields the realization

\[
\dot{x} = \begin{bmatrix}
0 & 0 & \cdots & 0 & -\alpha_0 \\
1 & 0 & \cdots & 0 & -\alpha_1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & -\alpha_{n-2} \\
0 & 0 & \cdots & 0 & 1 -\alpha_{n-1}
\end{bmatrix} x + \begin{bmatrix}
\beta_0 - \alpha_0\beta_n \\
\beta_1 - \alpha_1\beta_n \\
\vdots \\
\beta_{n-2} - \alpha_{n-2}\beta_n \\
\beta_{n-1} - \alpha_{n-1}\beta_n
\end{bmatrix} u
\]

\[
y = \begin{bmatrix}
0 & 0 & \cdots & 0 & 0 & 1
\end{bmatrix} \bar{x} + \beta_n u
\]

Although the MIMO OCF case is more involved, generating a MISO OCF is a straightforward generalization of the SISO case. For example if we have a two input transfer matrix

\[
G(s) = \frac{1}{s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1s^1 + \alpha_0} \begin{bmatrix}
\beta_{1n}s^n + \beta_{1n-1}s^{n-1} + \cdots + \beta_{10} \\
\beta_{2n}s^n + \beta_{2n-1}s^{n-1} + \cdots + \beta_{20}
\end{bmatrix}^T
\]

we get

\[
\dot{x} = \begin{bmatrix}
-\alpha_{n-1} & 1 & 0 & \cdots & 0 \\
-\alpha_{n-2} & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\alpha_1 & 0 & 0 & \cdots & 1 \\
-\alpha_0 & 0 & 0 & \cdots & 0
\end{bmatrix} x + \begin{bmatrix}
\beta_{1n-1} - \alpha_{n-1}\beta_{1n} & \beta_{2n-1} - \alpha_{n-1}\beta_{2n} \\
\beta_{1n-2} - \alpha_{n-2}\beta_{1n} & \beta_{2n-2} - \alpha_{n-2}\beta_{2n} \\
\vdots & \vdots \\
\beta_{11} - \alpha_1\beta_{1n} & \beta_{21} - \alpha_1\beta_{2n} \\
\beta_{10} - \alpha_0\beta_{1n} & \beta_{20} - \alpha_0\beta_{2n}
\end{bmatrix} u
\]

\[
y = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0
\end{bmatrix} x + \begin{bmatrix}
\beta_{1n} & \beta_{2n}
\end{bmatrix} u
\]

This idea easily generalizes to the \( m > 2 \) input case.

Jordan Canonical Form (JCF)

Another important canonical form is the Jordan Canonical Form (JCF) which results in a diagonal system matrix when the roots of the denominator polynomial are distinct. We
Figure 2.4: Simulation diagram of OCF (2.15). Summing junctions which are not labelled are positive.

Consider the SISO system (2.15) and first assume the poles of the system are distinct. Hence, (2.15) has a partial fraction expansion

\[ G(s) = \beta_n + \frac{r_1}{s - s_1} + \frac{r_2}{s - s_2} + \cdots + \frac{r_n}{s - s_n} \]

This expansion suggests a block diagram which consists of \( n \) first order systems in parallel with a direct path of gain \( \beta_n \). This block diagram is shown in Figure 2.5 where the residues are placed at the output of the integrators. (They could have alternatively been placed on the input side of the integrator).

Letting the outputs of the integrators be the states we have the following JCF realization equations

\[
\begin{align*}
\dot{x}_1 &= s_1 x_1 + u \\
\dot{x}_2 &= s_2 x_2 + u \\
&\vdots \\
\dot{x}_n &= s_n x_n + u \\
y &= r_1 x_1 + r_2 x_2 + \cdots + r_n x_n + \beta_n u
\end{align*}
\]
Figure 2.5: Jordan Canonical Form block diagram of (2.15) when poles are distinct. Summing junctions which are not labelled are positive.
Hence, the JCF is

\[
G = \begin{bmatrix}
    s_1 & 0 & 0 & \cdots & 0 \\
    0 & s_2 & 0 & \cdots & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 \cdots & s_n
\end{bmatrix} \begin{bmatrix}
    1 \\
    1 \\
    \vdots \\
    1
\end{bmatrix}u
\]

If \( G(s) \) has \( q < n \) distinct poles \( s_i, 1 \leq i \leq q \) with the multiplicity of \( s_i \) being \( m_i \) we break \( G(s) \) into \( q + 1 \) subsystems

\[
\frac{Y(s)}{U(s)} = G(s) = \beta_n + G_1(s) + \cdots + G_q(s)
\]

where the \( q \) subsystems \( G_i(s), 1 \leq i \leq q \) have transfer functions

\[
\frac{Y_i(s)}{U(s)} = G_i(s) = \frac{r_{1i}}{s-s_i} + \cdots + \frac{r_{mi}}{(s-s_i)^{m_i}}
\]

The realization of \( G_i(s) \) is shown in Figure 2.6 and using the integrator outputs as states its state space model is

\[
\dot{x}^i = \begin{bmatrix}
    s_i & 1 & 0 & \cdots & 0 & 0 & 0 \\
    0 & s_i & 1 & \cdots & 0 & 0 & 0 \\
    0 & 0 & s_i & \ddots & 0 & 0 & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
    0 & 0 & 0 & \cdots & s_i & 1 & 0 \\
    0 & 0 & 0 & \cdots & 0 & s_i & 1 \\
    0 & 0 & 0 & \cdots & 0 & 0 & s_i
\end{bmatrix} \begin{bmatrix}
    x^i \\
    0 \\
    0 \\
    \vdots \\
    0 \\
    0 \\
    1
\end{bmatrix} + \begin{bmatrix}
    u
\end{bmatrix}
\]

\[
y_i = \begin{bmatrix}
    r_{mi} & r_{m_i-1} & \cdots & r_{1i}
\end{bmatrix} \begin{bmatrix}
    x^i
\end{bmatrix}
\]

where \( x^i = (x_{1i}, x_{2i}, \ldots, x_{mi})^T \in \mathbb{R}^{m_i} \) is the state of subsystem \( G_i(s) \) and \( y_i \) is its output. The realization of the entire system \( G(s) \) is the parallel combination of the \( q \) subsystems. Hence, the JCF is

\[
\dot{x} = \text{blockdiag}\{A_1, A_2, \ldots, A_q\} \begin{bmatrix}
    x^1 \\
    x^2 \\
    \vdots \\
    x^q
\end{bmatrix} + \begin{bmatrix}
    B_1 \\
    B_2 \\
    \vdots \\
    B_q
\end{bmatrix}u
\]

\[
y = \begin{bmatrix}
    C_1 & C_2 & \cdots & C_q
\end{bmatrix} \begin{bmatrix}
    x^1 \\
    x^2 \\
    \vdots \\
    x^q
\end{bmatrix} + \begin{bmatrix}
    \beta_n \\
    \beta_{n-1} \\
    \vdots \\
    \beta_1
\end{bmatrix}u
\]
2.2. TRANSFER FUNCTION TO STATE SPACE (TF2SS)

Other system interconnection formula (series, feedback) will be seen below in Section 2.4.

Example 8 Consider the double integrator system \( Y(s)/U(s) = 1/s^2 \). Here, the CCF, OCF, and JCF are identical with
\[
A_c = A_o = A_d = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B_c = B_o = B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C_c = C_o = C_d = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D_c = D_o = D_d = 0
\]

□

Example 9 In this example we compute the CCF, OCF, and JCF for the DC motor
\[
\frac{Y(s)}{U(s)} = \frac{K_1}{s(s + K_2)}
\]

The entries of the CCF matrices can be simply read off the transfer function coefficients.

We have
\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -K_2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
\]
\[
y = \begin{bmatrix} K_1 & 0 \end{bmatrix} x
\]
Similarly the entries of OCF matrices can also be read off the transfer function coefficients. We have

\[
\dot{x} = \begin{bmatrix} -K_2 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ K_1 \end{bmatrix} u
\]

\[
y = \begin{bmatrix} 1 & 0 \end{bmatrix} x
\]

To obtain the JCF we perform a partial fraction expansion

\[
\frac{Y(s)}{U(s)} = \frac{K_1/K_2}{s} + \frac{-K_1/K_2}{s + K_2}
\]

Hence the JCF is

\[
\dot{x} = \begin{bmatrix} 0 & 0 \\ 0 & -K_2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u
\]

\[
y = \begin{bmatrix} K_1/K_2 & -K_1/K_2 \end{bmatrix} x
\]

\[\square\]

### 2.3 Linear change of state coordinates (ss2ss)

Given (2.1) we can define new state coordinates

\[
\xi = M x
\]

where \( M \in \mathbb{R}^{n \times n} \) is an invertible matrix. Taking the time derivative of both sides of (2.22) we obtain

\[
\frac{d\xi}{dt} = M \frac{dx}{dt} = M(Ax + Bu) = MAM^{-1}\xi + MBu
\]

and substituting \( x = M^{-1}\xi \) into (2.1b) gives

\[
y = CM^{-1}\xi + Du
\]

Hence the system in the \( \xi \)-coordinates is

\[
\dot{\xi} = \bar{A}\xi + \bar{B}u
\]

\[
y = \bar{C}x + \bar{D}u
\]

where

\[
\bar{A} = MAM^{-1}, \quad \bar{B} = MB, \quad \bar{C} = CM^{-1}, \quad \bar{D} = D
\]

When \( \bar{A} = MAM^{-1} \) we say \( A \) is similar to \( \bar{A} \).
When transforming state coordinates, confusion sometimes arises due to choice of notation. It should be noted that the use of symbols \( \xi, M \) in (2.22) is arbitrary. Any symbols could be
used instead (e.g. \( z, T \) in place of \( \xi, M \), respectively). Also, in some cases new \( z \)-coordinates are defined using \( zM = x \), and this alters the formulas for matrices in the new coordinates (2.25). As long as you work consistently, no confusion should arise with any choice of notation or definition of new coordinates. We remark that as there are an infinite number of invertible matrices which can be used to define an infinite number of state coordinate transformations. Hence, any given system has an \textit{infinite} number of realizations.

We remark that a change of state coordinates does not change the transfer matrix of a system. This can be seen by directly computing the transfer matrices in old and new state coordinates using (2.4). The transfer matrix of (2.1) is

\[
G_1(s) = C(sI - A)^{-1}B + D
\]  

The transfer matrix in the new state coordinates \( \xi = Mx \) is

\[
G_2(s) = \bar{C}(sI - \bar{A})^{-1}\bar{B} + \bar{D}
= CM^{-1}(sI - MAM^{-1})^{-1}MB + D
= CM^{-1}M(sI - A)^{-1}M^{-1}MB + D
= G_1(s)
\]

Above we have used the fact that for square nonsingular matrices \( P, Q \in \mathbb{R}^{n \times n} \) we have \((PQ)^{-1} = Q^{-1}P^{-1}\). The result that \( G_1(s) = G_2(s) \) is to be expected, as the system does not know which choice of state variables we have made and therefore this choice should not affect how the input determines the output.

\textbf{Example 10} Let’s reconsider the mass spring system in Example 3 and write its state space model in new state coordinates corresponding to the centre of mass of the system and the difference of the positions of the masses. These are commonly used coordinates when studying mass spring systems.

That is, the centre of mass is

\[
\bar{x} = \frac{m_1}{m_1 + m_2}y_1 + \frac{m_2}{m_1 + m_2}y_2
\]

and the difference in position is

\[
\delta = y_1 - y_2
\]

We define the new state as

\[
\xi = (\bar{x}, \dot{\bar{x}}, \delta, \dot{\delta})^T
\]

which means

\[
M = \begin{bmatrix}
\frac{m_1}{m_1 + m_2} & 0 & \frac{m_2}{m_1 + m_2} & 0 \\
0 & \frac{m_1}{m_1 + m_2} & 0 & \frac{m_2}{m_1 + m_2} \\
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1
\end{bmatrix}
\]
Using (2.25), in the ξ-coordinates the system has the form

\[
\dot{\xi} = \begin{bmatrix}
0 & 1 & 0 & 0 \\
-\frac{2k_1}{m_1+m_2} & 0 & \frac{k_1(m_1-m_2)}{(m_1+m_2)^2} & 0 \\
0 & 0 & 0 & 1 \\
\frac{k_1(m_1-m_2)}{m_1m_2} & \frac{(k_1+k_2)m_1^2+2m_1m_2k_2+m_2^2k_1+m_2^2k_2}{m_1m_2(m_1+m_2)} & 0 & 0
\end{bmatrix} \xi + \begin{bmatrix}
0 \\
\frac{1}{(m_1+m_2)} & 1 \\
0 & 1/(m_1) \\
1/m_1 & -1/m_2
\end{bmatrix} u
\]

\[
y = \begin{bmatrix}
1 & 0 & m_2/(m_1+m_2) & 0 \\
1 & 0 & -m_1/(m_1+m_2) & 0
\end{bmatrix} \begin{bmatrix}
\xi 
\end{bmatrix}
\]

Notice that for the case where the masses are only attached to each other (i.e., \(k_1 = 0\)), the dynamics become decoupled. That is, the solutions of \(\delta\) do not depend on \(\bar{x}\) and vice versa.

The MATLAB code is

```matlab
% Define symbolic variables
syms m1 m2 k1 k2
% Matrices of realization in original state coordinates
A=[0 1 0 0;... 
   -(k1+k2)/m1 0 k2/m1 0;... 
   0 0 0 1;... 
   k2/m2 0 -(k1+k2)/m2 0];
B=[0 0;1/m1 0;0 0;0 1/m2];
C=[1 0 0 0;0 0 1 0];
M=[m1/(m1+m2) 0 m2/(m1+m2) 0;0 m1/(m1+m2) 0 m2/(m1+m2);1 0 -1 0;0 1 0 -1] 
\%
disp('Abar')
pretty(simple(M*A*inv(M)))
disp('Bbar')
pretty(simple(M*B))
disp('Cbar')
pretty(simple(C*inv(M)))
```

2.4 System interconnections

It is often useful to interconnect two or more state space systems. The common interconnections are: serial, parallel, and feedback. These interconnections are shown in Figure 2.7.
We consider two systems
\[
\begin{align*}
\dot{x}_i &= A_i x_i + B_i u_i \\
y_i &= C_i x_i + D_i u_i \\
&\quad i = 1, 2
\end{align*}
\] (2.27)

**Series interconnections**

For the series configuration we have \( y = y_2, y_1 = u_2, u = u_1 \) so
\[
\begin{align*}
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} A_1 & 0 \\ B_2 C_1 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 D_1 \end{bmatrix} u \\
y &= \begin{bmatrix} D_2 C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D_2 D_1 u
\end{align*}
\]

This same result (with states reordered) can be obtained in MATLAB using
\[
s1=ss(A1,B1,C1,D1); \quad \% \text{Creates a state space model}
\]
\[
s2=ss(A2,B2,C2,D2);
\]
However, MATLAB only accepts numerical-valued (as opposed to symbolic-valued) $A$, $B$, $C$, $D$. We reconsider the example in Section 1.4 which considered the series interconnection of
\[
\frac{Y(s)}{U(s)} = P(s) = \frac{1}{s-1} \quad \text{and} \quad \frac{U(s)}{V(s)} = C(s) = \frac{U(s)}{V(s)} = \frac{s-1}{s+1}
\]
We take CCF realizations for $C(s)$ (System 1) and $P(s)$ (System 2) and apply the series formula to get a realization of the system with input $v$ and output $y$. We have
\[
A_1 = -1, B_1 = 1, C_1 = -2, D_1 = 1, \quad A_2 = 1, B_2 = 1, C_2 = 1, D_2 = 0
\]
Hence,
\[
\dot{x} = \begin{bmatrix} A_1 & 0 \\ B_2C_1 & A_2 \end{bmatrix} x + \begin{bmatrix} B_1 \\ B_2D_1 \end{bmatrix} v = \begin{bmatrix} -1 & 0 \\ -2 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \end{bmatrix} v
\]
\[
y = \begin{bmatrix} D_2C_1 & C_2 \end{bmatrix} x + D_2D_1 u = \begin{bmatrix} 0 & 1 \end{bmatrix} x
\]

**Parallel interconnections**

For the parallel configuration we have $y = y_1 + y_2$ and $u_1 = u_2 = u$ so that
\[
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u
\]
\[
y = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + (D_2 + D_1) u
\]

This same result (with states reordered) can be obtained in MATLAB using
\[
s_1 = \text{ss}(A_1, B_1, C_1, D_1);
\]
\[
s_2 = \text{ss}(A_2, B_2, C_2, D_2);
\]
\[
s_3 = s_2 + s_1;
\]

**Feedback interconnections**

For the feedback configuration we have $u_1 = v - y_2, y = y_1 = u_2$. We have
\[
u_1 = v - y_2
\]
\[
= v - (C_2 x_2 + D_2 u_2)
\]
\[
= v - (C_2 x_2 + D_2 (C_1 x_1 + D_1 u_1))
\]
\[
= v - (C_2 x_2 + D_2 C_1 x_1 + D_2 D_1 u_1)
\]

which implies
\[
(I + D_2 D_1) u_1 = v - C_2 x_2 - D_2 C_1 x_1
\]
\[
u_1 = (I + D_2 D_1)^{-1}(v - D_2 C_1 x_1 - C_2 x_2) = Q(v - D_2 C_1 x_1 - C_2 x_2)
\]
where \( Q = (I + D_2D_1)^{-1} \). Note that in the expression for \( Q \) the dimension of \( I \) must be the same as \( D_2D_1 \) which is a \( p_2 \times m_1 \) matrix, where \( p_2 = \dim(y_2), m_1 = \dim(u_1) \). Hence
\[
\dot{x}_1 = A_1x_1 + B_1u_1 = A_1x_1 + B_1Q(v - D_2C_1x_1 - C_2x_2)
\]
\[
= (A_1 - B_1QD_2C_1)x_1 - B_1QC_2x_2 + B_1Qv
\]
and
\[
\dot{x}_2 = A_2x_2 + B_2u_2 = A_2x_2 + B_2y_1 = A_2x_2 + B_2(C_1x_1 + D_1u_1)
\]
\[
= A_2x_2 + B_2C_1x_1 + B_2D_1Q(v - D_2C_1x_1 - C_2x_2)
\]
\[
= (A_2 - B_2D_1QC_2)x_2 + (B_2C_1 - B_2D_1QD_2C_1)x_1 + B_2D_1Qv
\]
For the output we have
\[
y = y_1 = C_1x_1 + D_1u_1 = C_1x_1 + D_1Q(v - D_2C_1x_1 - C_2x_2)
\]
\[
= (C_1 - D_1QD_2C_1)x_1 - D_1QC_2x_2 + D_1Qv
\]
In matrix form
\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix}
= 
\begin{bmatrix}
A_1 - B_1QD_2C_1 & -B_1QC_2 \\
B_2C_1 - B_2D_1QD_2C_1 & A_2 - B_2D_1QC_2
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
+ 
\begin{bmatrix}
B_1Q \\
B_2D_1Q
\end{bmatrix}
v
\]
\[
y = 
\begin{bmatrix}
C_1 - D_1QD_2C_1 & -D_1QC_2
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
+ D_1Qv
\]
This same result (with states reordered) can be obtained in MATLAB using
\[
s1=ss(A1,B1,C1,D1);
\]
\[
s2=ss(A2,B2,C2,D2);
\]
\[
s3=feedback(s1,s2);
\]
Note that the MATLAB command \texttt{sysic} can be used to create very general system interconnections.

## 2.5 Linearization of nonlinear state space systems

Many real systems (e.g. the ROTPEN in the lab) are accurately modelled using a nonlinear time-invariant state space model given by
\[
\dot{x} = f(x, u) \quad \text{(2.28a)}
\]
\[
y = h(x, u). \quad \text{(2.28b)}
\]
Here we treat the MIMO case: \( y \in \mathbb{R}^p, u \in \mathbb{R}^m, x \in \mathbb{R}^n \) and we assume that vector-valued functions \( f \) and \( h \) are smooth. Note that we have ignored any disturbance inputs only to simplify presentation. These could also be included by treating them analogously to input \( u \).

Although many systems are accurately modelled using (2.28), the nonlinearity of these systems can make analysis or controller design difficult. Consequently, we often compute an approximate linear system or so-called \textit{linearization} which can:
• provide a good local description of the original nonlinear system’s behaviour,
• be more amenable to analysis and control design methods due to its linearity.

The method to obtain this linearization relies on a function’s Taylor series expansion. We consider two types of linearizations:

**Case 1:** Linearization about a general (possibly non-constant) nominal state and input trajectory denoted \((x_r(t), u_r(t)), t \geq 0\).

**Case 2:** Linearization about a constant *equilibrium* trajectory \((x_e(t), u_e(t)) = \text{const.}, t \geq 0\) such that \(f(x_e, u_e) = 0\).

We remark that Case 2 is a special case of Case 1.

**Case 1: Linearization about a nominal trajectory**

Given some nominal input function \(u_r(t), t \geq 0\), we can compute the system’s corresponding state response, denoted by \(x_r(t), t \geq 0\), and output response, denoted by \(y_r(t), t \geq 0\), using \((2.28)\) for a given initial condition \(x_r(0) = x_{0r}\). That is, the trajectories \(x_r(t)\) and \(y_r(t)\) necessarily satisfy the ODE \((2.28a)\) and output equation \((2.28b)\):

\[
\dot{x}_r = f(x_r, u_r) \tag{2.29a}
\]
\[
y_r = h(x_r, u_r) \tag{2.29b}
\]

Next we expand the functions \(f\) and \(h\) about the nominal trajectory \((x_r(t), u_r(t))\) using a Taylor series

\[
f(x, u) = f(x + \Delta x, u + \Delta u) = f(x_r, u_r) + \frac{\partial f}{\partial x}(x_r, u_r)\Delta x + \frac{\partial f}{\partial u}(x_r, u_r)\Delta u + \text{H.O.T.} \tag{2.30a}
\]
\[
h(x, u) = h(x + \Delta x, u + \Delta u) = h(x_r, u_r) + \frac{\partial h}{\partial x}(x_r, u_r)\Delta x + \frac{\partial h}{\partial u}(x_r, u_r)\Delta u + \text{H.O.T.} \tag{2.30b}
\]

where “H.O.T.” denotes higher order terms in \(\Delta x\) and \(\Delta u\), we have defined the deviation variables \(\Delta x = x - x_r\), \(\Delta u = u - u_r\), and we have used the standard notation

\[
\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}, \quad \frac{\partial f}{\partial u} = \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{pmatrix}
\]
\[
\frac{\partial h}{\partial x} = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_n} \end{pmatrix}, \quad \frac{\partial h}{\partial u} = \begin{pmatrix} \frac{\partial h_1}{\partial u_1} & \cdots & \frac{\partial h_p}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1}{\partial u_1} & \cdots & \frac{\partial h_p}{\partial u_m} \end{pmatrix}
\]

The \(n \times n\) matrix \(\frac{\partial f}{\partial x}\), \(n \times m\) matrix \(\frac{\partial f}{\partial u}\), the \(p \times n\) matrix \(\frac{\partial h}{\partial x}\), and \(p \times m\) matrix \(\frac{\partial h}{\partial u}\) are called Jacobian matrices.
Hence subbing (2.30) into (2.28) we have

\[ \dot{x}_r + \Delta \dot{x} = f(x_r, u_r) + \frac{\partial f}{\partial x}(x_r, u_r) \Delta x + \frac{\partial f}{\partial u}(x_r, u_r) \Delta u \]  

(2.31a)

\[ y_r + \Delta y = h(x_r, u_r) + \frac{\partial h}{\partial x}(x_r, u_r) \Delta x + \frac{\partial h}{\partial u}(x_r, u_r) \Delta u \]  

(2.31b)

where we have assumed “H.O.T.” are negligible since \( x \approx x_r, u \approx u_r \) and have defined \( \Delta y = y - y_r \). The cancelation in (2.31) follows from (2.29).

Rewriting (2.31) gives

\[ \Delta \dot{x} = \frac{\partial f}{\partial x}(x_r, u_r) \Delta x + \frac{\partial f}{\partial u}(x_r, u_r) \Delta u \]  

(2.32a)

\[ \Delta y = \frac{\partial h}{\partial x}(x_r, u_r) \Delta x + \frac{\partial h}{\partial u}(x_r, u_r) \Delta u \]  

(2.32b)

Defining the time-varying matrices \( A(t), B(t), C(t), D(t) \) in (2.32), we have a linear time-varying state space system in the deviation variables

\[ \Delta \dot{x} = A(t) \Delta x + B(t) \Delta u \]  

(2.33a)

\[ \Delta y = C(t) \Delta x + D(t) \Delta u \]  

(2.33b)

It is important to note that the approximate linear system (2.33) is in terms of deviation variables \( \Delta x, \Delta u, \) and \( \Delta y \) and not the original variables \( x, u, \) and \( y \).

### Case 2: Linearization about an equilibrium solution

Often we are interested in approximating a nonlinear system (2.28) about a so-called equilibrium point\(^2\), i.e., a constant state \( x_e \) and a corresponding constant input \( u_e \) which satisfy

\[ f(x_e, u_e) = 0 \]  

(2.34)

It is important to note that in general for systems (2.28), there may be an infinite number of equilibrium points, or there may be none. Also, simply taking \( u_e \) constant does not ensure a constant state trajectory. However, for many real-world systems there is at least one equilibrium point of practical importance where we can hold \( y \) and \( x \) constant for a given constant \( u \).

Using Taylor series to expand functions \( f \) and \( h \) about the constant value \( (x_e, u_e) \) gives

\[ f(x_e + \Delta x, u_e + \Delta u) = f(x_e, u_e) + \frac{\partial f}{\partial x}(x_e, u_e) \Delta x + \frac{\partial f}{\partial u}(x_e, u_e) \Delta u + \text{HOT} \]  

(2.35a)

\[ h(x_e + \Delta x, u_e + \Delta u) = h(x_e, u_e) + \frac{\partial h}{\partial x}(x_e, u_e) \Delta x + \frac{\partial h}{\partial u}(x_e, u_e) \Delta u + \text{HOT} \]  

(2.35b)

\(^2\)sometimes referred to as a constant operating point
Similar to (2.31), we obtain

\[\begin{align*}
\dot{x}_e + \Delta \dot{x} &= f(x_e, u_e) + \frac{\partial f}{\partial x}(x_e, u_e) \Delta x + \frac{\partial f}{\partial u}(x_e, u_e) \Delta u \\
y_e + \Delta y &= h(x_e, u_e) + \frac{\partial h}{\partial x}(x_e, u_e) \Delta x + \frac{\partial h}{\partial u}(x_e, u_e) \Delta u
\end{align*}\]  
(2.36a)

Rewriting (2.31) gives

\[\begin{align*}
\Delta \dot{x} &= \frac{\partial f}{\partial x}(x_e, u_e) \Delta x + \frac{\partial f}{\partial u}(x_e, u_e) \Delta u \\
\Delta y &= \frac{\partial h}{\partial x}(x_e, u_e) \Delta x + \frac{\partial h}{\partial u}(x_e, u_e) \Delta u
\end{align*}\]  
(2.37a)

Defining the constant matrices \( A, B, C, D \) in (2.37) we have a LTI system in the deviation variables

\[\begin{align*}
\Delta \dot{x} &= A \Delta x + B \Delta u \\
\Delta y &= C \Delta x + D \Delta u
\end{align*}\]  
(2.38a)

The linearized system (2.38) is the same as that derived using different notation in [Bay99, Eqn. (1.62)].

**Example 11 Case 1: Simple System** Consider a nonlinear state space system model

\[\begin{align*}
\dot{x} &= f(x, u) = -ax + bu \\
y &= x
\end{align*}\]

Suppose we take a constant nominal input \( u_r(t) = a/(2b), t \geq 0 \). This nominal input results in the nominal state trajectory

\[x_r(t) = x_{0r}e^{-at/2}\]  
(2.39)

for some initial condition \( x_{0r} \). As \( x_r(t) \) varies with time, it is not an equilibrium solution even though we took a constant input. It is worth noting that in general for nonlinear systems, we cannot always obtain closed-form expressions such as (2.39) for the state for a given input and initial state. We have

\[\begin{align*}
\frac{\partial f}{\partial x} &= -a + bu, & \frac{\partial f}{\partial u} &= bx, & \frac{\partial h}{\partial x} &= 1
\end{align*}\]

Hence from (2.32) we have

\[\begin{align*}
A(t) &= \frac{\partial f}{\partial x}(x_r, u_r) = -a/2, & B(t) &= \frac{\partial f}{\partial u}(x_r, u_r) = bx_{0r}e^{-at/2}, & C(t) &= \frac{\partial h}{\partial x}(x_r, u_r) = 1
\end{align*}\]  
(2.40)
The linearized system about the nominal trajectory is therefore

\[ \Delta \dot{x} = A(t) \Delta x + B(t) \Delta u \] (2.41a)
\[ \Delta y = C(t) \Delta x \] (2.41b)

where \( A(t), B(t), C(t) \) are defined in (2.40) and

\[ \Delta x = x - x_r, \quad \Delta u = u - a/(2b), \quad \Delta y = y - x_r \]

\[ \Box \]

**Example 12 Case 1: Rocket** The height \( x_1 \), velocity \( x_2 \), and mass \( x_3 \) of a rocket are modelled by

\[
\begin{align*}
\dot{x}_1 &= x_2 \quad \text{(2.42a)} \\
\dot{x}_2 &= -g + au/x_3 \quad \text{(2.42b)} \\
\dot{x}_3 &= u \quad \text{(2.42c)} \\
y &= x_1 \quad \text{(2.42d)}
\end{align*}
\]

The SISO system input is the time derivative of the mass (i.e., \( u = \dot{x}_3 \)) and the output is the height of the rocket \( y = x_1 \). The acceleration due to gravity is denoted \( g \), and \( a \) is a constant parameter. Clearly the state space model (2.42) is nonlinear and of the form (2.28) with

\[
f(x, u) = \begin{pmatrix} x_2 \\ -g + au/x_3 \\ u \end{pmatrix}, \quad h(x, u) = x_1
\]

(2.43)

We consider the linearization of (2.42) about a nominal trajectory (i.e., Case 1) corresponding to a constant input \( u_r(t) = u_0 < 0, t \geq 0 \) and initial state \( x_r(0) = (0, 0, x_{30r})^T \). We can integrate (2.42) in closed-form for this input and initial state to obtain the nominal state trajectory \( x_r = (x_{1r}, x_{2r}, x_{3r})^T \). We obtain

\[
\begin{align*}
x_{1r} &= -\frac{gt^2}{2} + \frac{x_{30r}a}{u_0} \left( 1 + \frac{u_0}{x_{30r}} t \right) \ln \left( 1 + \frac{u_0}{x_{30r}} t \right) - \frac{u_0}{x_{30r}} t \\
x_{2r} &= -gt + a \ln \left( 1 + \frac{u_0}{x_{30r}} t \right) \\
x_{3r} &= x_{30r} + u_0 t
\]
\]

The Jacobian matrices are

\[
\frac{\partial f}{\partial x} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -au/(x_3^2) \\ 0 & 0 & 0 \end{pmatrix}, \quad \frac{\partial f}{\partial u} = \begin{pmatrix} 0 \\ a/x_3 \\ 1 \end{pmatrix}, \quad \frac{\partial h}{\partial x} = (1 \ 0 \ 0)
\]
Hence we have

\[
A(t) = \frac{\partial f}{\partial x}(x_r, u_r) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -au/(x_{30r} + u_0t)^2 \\ 0 & 0 & 0 \end{pmatrix}
\]

\[
B(t) = \frac{\partial f}{\partial u}(x_r, u_r) = \begin{pmatrix} 0 \\ a/(x_{30r} + u_0t) \\ 1 \end{pmatrix}
\]

\[
C(t) = \frac{\partial h}{\partial x}(x_r, u_r) = (1 \ 0 \ 0)
\]

Hence the linearized system about the nominal trajectory is therefore

\[
\Delta \dot{x} = A(t)\Delta x + B(t)\Delta u 
\]
\[
\Delta y = C(t)\Delta x
\]

where \(A(t), B(t), C(t)\) are defined in (2.45) and

\[
\Delta x = x - x_r, \quad \Delta u = u - u_0, \quad \Delta y = y - x_{1r}
\]

**Example 13 Case 2: Cart and Pendulum** Consider a cart and pendulum system shown in Figure 2.8. Here an input force \(u\) is applied to a cart of mass \(M\) to balance a pendulum of length \(l\) in its upright position. The pendulum is massless and has a point mass \(m\) fixed to its end. The output of the system is the cart’s position and pendulum angle. Note the strong similarity of this system with the ROTPEN which is essentially a “rotational version” of this
system. We wish to compute the linearization of this system about one of its equilibrium points (i.e., Case 2).

The nonlinear ODEs for the pendulum on a cart are given by

\[ \begin{align*}
M \ddot{y}_1 &= u - m \ddot{y}_1 - ml \ddot{\theta} \cos \theta + ml(\dot{\theta})^2 \sin \theta \\
g \sin \theta &= l \ddot{\theta} + \ddot{y}_1 \cos \theta
\end{align*} \quad (2.47a) \]

Using the states defined as

\[ x_1 = y_1, \quad x_2 = \dot{y}_1, \quad x_3 = \theta, \quad x_4 = \dot{\theta} \]

we put the system in state space form

\[ \dot{x} = f(x, u), \quad y = h(x, u). \]

We rewrite (2.47) in matrix form

\[ \begin{pmatrix} M + m & ml \cos x_3 \\ \cos x_3 & l \end{pmatrix} \begin{pmatrix} \dot{x}_2 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_4^2 ml \sin x_3 + u \\ g \sin x_3 \end{pmatrix} \]

\[ \begin{pmatrix} \dot{x}_2 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} M + m & ml \cos x_3 \\ \cos x_3 & l \end{pmatrix}^{-1} \begin{pmatrix} x_4^2 ml \sin x_3 + u \\ g \sin x_3 \end{pmatrix} \]

\[ \begin{pmatrix} \dot{x}_2 \\ \dot{x}_4 \end{pmatrix} = \frac{1}{M + m \sin^2 x_3} \begin{pmatrix} u + mlx_4^2 \sin x_3 - mg \sin x_3 \cos x_3 \\ ((M + m)g \sin x_3 - mlx_4^2 \sin x_3 \cos x_3 - u \cos x_3)/l \end{pmatrix} \]

Hence, the state space form of the system is

\[ \begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= (u + mlx_4^2 \sin x_3 - mg \sin x_3 \cos x_3)/(M + m \sin^2 x_3) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= ((M + m)g \sin x_3 - mlx_4^2 \sin x_3 \cos x_3 - u \cos x_3)/(l(M + m \sin^2 x_3)) \quad (2.48d)
\end{align*} \]

\[ y = \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} \]

**Equilibrium Set** Since the equilibrium state \( x_e \) and equilibrium input \( u_e \) must satisfy \( f(x_e, u_e) = 0 \) we have immediately \( x_2e = 0, x_4e = 0 \) from (2.48). As well, we immediately conclude that \( x_{1e} = \text{const.} \) (i.e., \( x_{1e} \) can be any constant value) given that \( f \) is independent of \( x_1 \). Finally, from (2.48b) we have

\[ u_e = mg \sin x_{3e} \cos x_{3e} \]

and from (2.48d) we have

\[ (M + m)g \sin x_{3e} = u_e \cos x_{3e} \]

Subbing (2.49) into (2.50) we have

\[ \frac{M + m}{m} \sin x_{3e} = \sin x_{3e} \cos^2 x_{3e} \]

which is satisfied iff \( x_{3e} = k\pi, k \in \mathbb{Z} \). This is because if \( x_{3e} \neq k\pi \) then we can cancel \( \sin x_3 \) from both sides of (2.51) and we obtain a relation which is never satisfied:

\[ \frac{M + m}{m} = \cos^2 x_{3e} \]
since \((M + m)/m > 1\). Hence, from (2.49) we have \(u_e = 0\). Hence the equilibrium set is given by
\[
\{(x_e, u_e) : x_{2e} = x_{4e} = 0, x_{1e} = \text{const.}, x_{3e} = k\pi, k \in \mathbb{Z}\}
\] (2.52)

The set (2.52) has physical significance as it indicates that the only way \(x = \text{const}\) is if the pendulum is the pendant or inverted position, and that these two configurations require a zero force input to remain there. A similar situation arises in the lab with the ROTPEN.

**Linearized system about equilibrium point** We compute the linearization of the system about \(x_e = (x_{1e} 0 0 0)^T, u_e = 0\) \((x_{1e}\) is any constant) by applying (2.37).

With little computation we have
\[
\frac{\partial f_1}{\partial x} = (0 1 0 0), \quad \frac{\partial f_3}{\partial x} = (0 0 0 1)
\] (2.53)

Straightforward but tedious calculation shows
\[
\frac{\partial f_2}{\partial x} = \begin{pmatrix} 0 & 0 & m(l^2 x_3 + g(\sin^2 x_3 - \cos^2 x_3)) & -2m \sin x_3 \cos x_3 (u + mlx_3^2 \sin x_3 \cos x_3) \\ 0 & 0 & (M+m) \sin^2 x_3 & 2mlx_3 \sin x_3 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix}
\] (2.54)

Evaluated at \((x_e, u_e)\) gives
\[
\frac{\partial f_2}{\partial x}(x_e, u_e) = (0 0 -mg/M 0)
\] (2.55)

Further we have
\[
\frac{\partial f_4}{\partial x} \begin{pmatrix} (M+m) \cos x_3 + mlx_3^2 (\sin^2 x_3 - \cos^2 x_3) + u \sin x_3 \\ l(M+m) \sin^2 x_3 \\ -2mx_3 \sin x_3 \cos x_3 \\ M+m \sin^2 x_3 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\] (2.56)

Evaluated at \((x_e, u_e)\) gives
\[
\frac{\partial f_4}{\partial x}(x_e, u_e) = (0 0 (M + m)g/(lM) 0)
\] (2.57)

Hence we have the linearized system
\[
\Delta \dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -mg/M & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & (M + m)g/(lM) & 0 \end{pmatrix} \Delta x + \begin{pmatrix} 0 \\ 1/M \\ 0 \\ -1/(lM) \end{pmatrix} u
\] (2.58)

\[
\Delta y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \Delta x
\] (2.59)

We remark that in this case \(\Delta x = (x_1 - x_{1e}, x_2, x_3, x_4)^T, \Delta y = (x_1 - x_{1e}, x_3)^T, \) and \(\Delta u = u.\)
MATLAB code to perform linearization  Computing the Jacobian matrices can be very tedious, fortunately MATLAB provides an easy solution with the Symbolic Math Toolbox. The code to generate the linearization can be downloaded at [www.ece.ualberta.ca/~alanl/ee460/x/examples/cart_pend_eg.m](http://www.ece.ualberta.ca/~alanl/ee460/x/examples/cart_pend_eg.m) and is given here

\begin{verbatim}
% EE 460 - Linearization of Cart Pendulum using Symbolic Math Toolbox

syms m M L g positive; syms x1e; syms x1 x2 x3 x4 u

x=[x1;x2;x3;x4];
x_u=[x1,x2,x3,x4,u];

% Equilibrium state (inverted configuration) and corresponding constant input
xe_ue=[x1e,0,0,0,0];

% Nonlinear state space form
f=[x2;...
   (u+m*L*x4^2*sin(x3)-m*g*sin(x3)*cos(x3))/(M+m*sin(x3)^2);...
   x4;...
   ((M+m)*g*sin(x3)-m*L*x4^2*sin(x3)*cos(x3)-u*cos(x3))/(L*(M+m*sin(x3)^2))];

h=[x1;x3];

% Jacobian matrices
dfdx=jacobian(f,x); dfdu=jacobian(f,u);

dhdx=jacobian(h,x); dhdu=jacobian(h,u);

% Evaluate Jacobian matrices
A=subs(dfdx,x_u,xe_ue);
B=subs(dfdu,x_u,xe_ue);
C=subs(dhdx,x_u,xe_ue);
D=subs(dhdu,x_u,xe_ue);

disp('Linearized system matrices');

A
B
C
D

% Compute transfer matrix of linearized system
syms s;

% Output is y = [x1;x3]
C2=[1 0 0 0;0 0 1 0];
\end{verbatim}
This program also computes the transfer matrix of the linearized system.
Bibliography

